BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

Citra digital merupakan citra yang telah disimpan dalam bentuk file sehingga dapat diolah dengan menggunakan komputer. Citra digital digunakan dalam berbagai bidang yang dapat membantu manusia dalam bekerja. Dalam penggunaan citra, tidak semua gambar digunakan, kadang-kadang hanya sebagian saja, membutuhkan beberapa perubahan seperti mengubah ukuran citra, mengubah tingkat kecerahan, serta menggabungkan dua citra atau lebih, proses tersebut biasanya disebut pengolahan citra (T, Sutoyo, 1999).

Pengolahan citra memiliki berbagai macam jenis klasifikasi. Salah satunya adalah segmentasi citra. Segmentasi citra merupakan suatu proses memecah suatu citra digital menjadi banyak segmen/bagian daerah yang tidak saling bertabrakan (*nonoverlapping*) dalam konteks citra digital daerah hasil segmentasi tersebut merupakan kelompok piksel yang bertetangga atau berhubungan (Castleman, 1996).

Terdapat berbagai macam metode segmentasi citra misalnya *thresholding* (Al-amri et al., 2010), *clustering* (Chitade dan Katiyar, 2010; Halder dan Pathak), *compression based* (Ni et al, 2009), *color image-based* (Bhoyar dan Kakde), *edge detection* (Senthilkumaran dan Rajesh, 2009) , *partial differential equation-based* (Wang et al, 2012;Peng et al, 1999), *waterhed transformation* (Vashist dan

Hema, 2013), *model based segmentation* (Bourouis dan Hamrouni, 2011), *active contour-model* (Airouche et al, 2009) dan lain-lain.

Active Contour merupakan salah satu algoritma yang digunakan dalam segmentasi citra. Kass et al (1987) menyatakan a*ctive contour* merupakan suatu batas objek atau beberapa fitur gambar lainnya sebagai kurva *parametric* (Kass et al, 1987). *Active contour* diimplementasikan dengan sebuah fungsi energi E yang dikaitkan dengan kurva dan memiliki variabel pengkoreksian (*error*) apabila terjadi kesalahan, maka didapatkan hasil yang akurat.

Active contour juga dapat digunakan untuk mensegmentasi citra menjadi beberapa *region* (Amine dan Farida, 2012). *Active contour* banyak diimplementasi pada berbagai macam citra, salah satunya untuk mendeteksi citra (Rocca et al). *Active Contour* juga dapat dikombinasikan dengan beberapa metode, salah satunya adalah *level set*. *Level set* merupakan salah satu metode yang menggunakan fungsi evolusi energi, maka dapat mengoptimalkan proses segementasi.

Brox dan Weickert (2004) menyatakan bahwa formula pada *level set Mumforh and Shah* terdapat beberapa kesalahan, yang diimplementasikan pada citra MRI otak dengan intensitas inhomogen. Berdasarkan permasalahan tersebut maka dilakukan penelitian yang mengembangkan formula *level set* untuk segmentasi citra pada citra yang memiliki intensitas inhomogen (Brox dan Weickert, 2004).

Li et al (2005) menyatakan bahwa fungsi *level set* yang digunakan memiliki kelemahan yaitu kurang optimal dalam proses komputasi. Berdasarkan

permasalahan tersebut maka dilakukan penelitian yang mengembangkan fungsi *level set* dengan menghilangkan proses reinisialisasi, dapat mempercepat proses evolusi *level set* (Li et al, 2005). Li et al (2007) mengembangkan fungsi *level set* menjadi LBF (*Local Binary Fitting*) model (Li et al, 2007).

Li et al (2010) mengembangkan penelitiannya pada fungsi *level set* yang memiliki penyimpangan pada proses evolusinya sehingga dapat mengakibatkan terjadi kesalahan dalam perhitungan numerik dan dapat merusak stabilitas dari evolusi. Untuk mengatasi masalah tersebut fungsi *level set* dikembangkan menjadi *Distance Regularized Level Set Evoluiton (DRLSE)*. DRSLE merupakan formula baru yang memiliki keteraturan secara intrinsik dalam evolusinya (Li et al, 2010).

Pada bidang kedokteran, tidak semua citra digunakan, tetapi hanya bagian tertentu yang dibutuhkan. *Level set* banyak digunakan dalam berbagai macam citra seperti citra tumor (Chen, 2008; Gao et al, 2012), citra sel-sel mikroskop (Yu et al, 2008), citra MRI (Barman et al, 2011), citra *Syntethic Aperture Image* (Ayed et al, 2005), *positronemission tomography* (Chan et al, 2007). Citra MRI memiliki kelemahan, yaitu tingkat intensitas pixel yang inhomogen, sehingga untuk mendapatkan keakuratan segmentasi citra sangat sulit. Maka Li et al (2011) mengembangkan *level set chan vese model* agar dapat diterapkan pada citra MRI yang memiliki intensitas inhomogen (Li et al, 2011).

Dengan berkembangnya teknologi dan ilmu pengetahuan, terutama dalam bidang pengolahan citra, permasalahan yang ditemukan semakin kompleks, maka membutuhkan tingkat keakuratan yang tinggi dalam proses pengolahan citra. Beberapa peneliti melakukan penelitian dengan menggabungkan *active contour*

model dan *level set method*. Kombinasi metode *level set* dan *active contour model* dapat diimplementasikan pada berbagai macam citra, seperti ISL (*Indian Sign Language*) (Krishnaveni dan Radha, 2011), citra MRI (Lazrag dan Naceur, 2012 ; Angelini et al, 2007).

Chan dan Zhu (2005) menyatakan bahwa pada proses penyebaran *level set* tidak memiliki acuan, maka dilakukan penelitian dengan menggabungkan *Chan-Vese Model* dengan *Shape* (Chan dan Zhu, 2005). Lee (2005) mengembangkan memperhalus hasil segmentasi citra, maka ditambahkan Gaussian Filter (Lee, 2005). Munim dan Abd (2007) bahwa *Shape* menjadi acuan proses penyebaran *level set* (Munim dan Abd, 2007). Dubrovina dan Kimmel (2012) mengembangkan *active contour model* dan *level set Mumford Shah Model* yang dikombinasikan untuk menyelesaikan masalah segementasi agar dapat disegmentasi citra menjadi beberapa region (Dubrovina dan Kimmel, 2012).

Zhang et al (2009) mengkaji masalah citra yang memiliki intensitas inhomogen, kadang-kadang hasil segementasi citra yang diperoleh kurang halus dan terlihat kaku. Berdasarkan masalah tersebut maka dilakukan penelitian yang mengembangkan kombinasi fungsi *level set method* dengan *active contour model* yang berfungsi untuk menghaluskan evolusi *level set*. Yang dikembangkan dengan meregularisasi fungsi *level set* menggunakan *Gaussian Filter* (Zhang et al, 2009).

Dengan mengoptimalisasi fungsi-fungsi atau metode–metode yang digunakan untuk mensegmentasi citra masih memiliki kelemahan, yaitu kurang efisiennya waktu yang digunakan. Proses segmentasi citra memiliki proses

komputasi yang kompleks, maka membutuhkan waktu yang cukup lama untuk menyelesaikan proses tersebut. Untuk mengatasi kelemahan tersebut ditemukan solusi yang baik, yaitu penggunaan komputasi paralel yang berbasis GPU (*Graphic Processing Unit*) CUDA.

GPU merupakan suatu perangkat grafis yang digunakan untuk kalkulasi grafis. Dengan *processor* parallel dapat membantu mempercepat proses komputasi, khususnya pada pengolahan citra (Ghorpade et al, 2012 ; Bhatotia et al ; Lin et al, 2012 ; Grama et al, 2003).

Adapun beberapa masalah yang diselesaikan menggunakan GPU, seperti enkripsi menggunakan AES yang diparalel menggunakan CUDA, dan memperoleh hasil transfer data 20x lebih cepat dibandingkan menggunakan CPU (Iwai et al., 2012), enkripsi dan deskripsi menggunakan *System P* menggunakan GPU menghasil kecepatan yang lebih cepat dibandingkan menggunakan CPU (Zaher et al., 2012), deteksi tepi citra menggunakan *canny method* berbasis GPU CUDA menghasilkan efisiensi waktu pada proses komputasi (Roodt et al., 2007), segmentasi citra biomedikal berbasis GPU yang menghasilkan kecepatan proses komputasi 30 sampai 40 kali lebih cepat (Mostofi, 2009), Optimasi ICA dan EEG (Raimondo et al., 2012), *Pattern matching* menggunakan metode Aho-Corasick hanya membutuhkan waktu 14.74 detik lebih cepat daripada CPU (Lin et al., 2012), pencitraan USG menggunakan *Angular Spectrum Method* (Varray et al., 2011), pengolahan citra menggunakan metode *fast fourrier transformation* (Haque dan Uddin, 2011).

2.2. Landasan Teori

2.2.1. Citra Digital

Menurut Sutoyo (2009) menyatakan bahwa citra merupakan suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra dibagi menjadi 2, yaitu

Citra Analog

Citra analog merupakan citra yang bersifat kontinyu. Misalnya gambar pada monitor televisi, dll.

Citra Digital

Citra digital merupakan citra yang dapat diolah oleh komputer.

Sebuah citra digital dapat mewakili oleh sebuah matriks yang terdiri dari M kolom N baris, dimana perpotongan antara kolom dan baris disebut piksel (piksel = *picture element*), yaitu elemen terkecil dari sebuah citra. Piksel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah f(x,y), yaitu besar intensitas atau warna dari piksel di titik itu.

Oleh sebab itu, sebuah citra digital dapat ditulis dalam bentuk matriks 2.1 berikut :

$$f(x,y) = \begin{matrix} f(0,0) & \dots & f(0,M-1) \\ \dots & \dots & f(1,M-1) \\ f(N-1,0) & f(N-1,1) & f(N-1,M-1) \end{matrix} \quad (2.1)$$

Berdasarkan gambaran tersebut, secara matematis citra digital dapat dituliskan sebagai fungsi intensitas f(x,y), dimana harga x (baris) dan y (kolom) merupakan koordinat posisi dan f(x,y) adalah nilai fungsi pada setiap titik (x,y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari piksel di titik tersebut. Pada proses digitalisasi (sampling dan kuantitas) diperoleh besar baris M

dan kolom N hingga citra membentuk matriks ***M x N*** dan jumlah tingkat keabuan piksel G (Sutoyo *et al*,2009).

2.2.2. Segmentasi Citra

Segmentasi citra akan membagi-bagi suatu citra menjadi daerah-daerah atau obyek-obyek yang dimilikinya. Menurut Castleman (1996) menyatakan bahwa segmentasi citra merupakan suatu proses memecah suatu citra digital menjadi banyak segmen/bagian daerah yang tidak saling bertabrakan (*nonoverlapping*) dalam konteks citra digital daerah hasil segmentasi tersebut merupakan kelompok piksel yang bertetangga atau berhubungan.

Segmentasi citra dapat dilakukan melalui beberapa pendekatan. Menurut Castleman (1996) menyatakan bahwa terdapat 3 macam pendekatan, antara lain :

a. Pendekatan batas (*boundary approach*)

   Pendekatan ini dilakukan untuk mendapatkan batas yang ada antar daerah.

b. Pendekatan tepi (*edge approach*)

   Pendekatan tepi dilakukan untuk mengidentifikasi piksel tepi dan menghubungkan piksel-piksel tersebut menjadi suatu batas yang diinginkan.

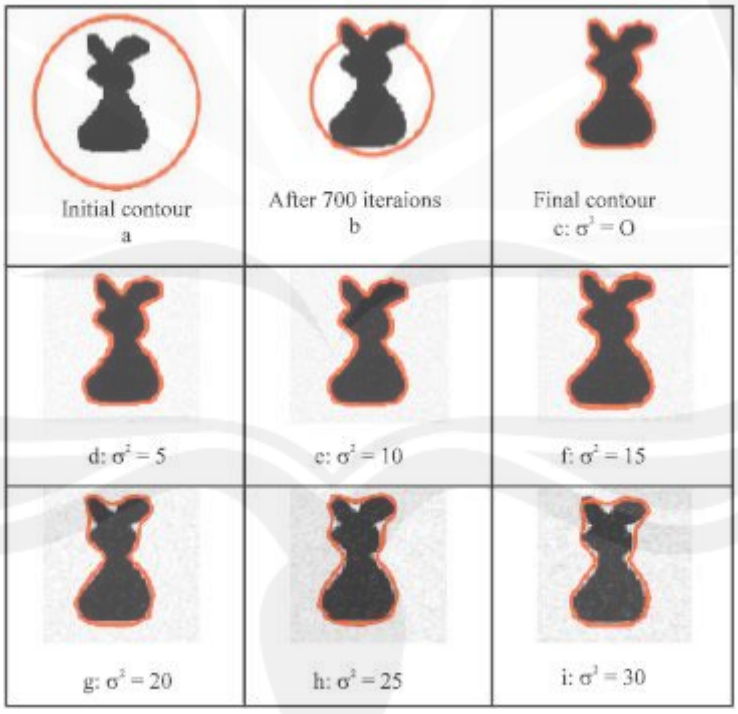
c. Pendekatan daerah (*region approach*)

   Pendekatan daerah bertujuan untuk membagi citra dalam daerah-daerah sehingga didapatkan suatu daerah sesuai kriteria yang diinginkan.

Proses segmentasi digunakan dalam berbagai penerapan, meskipun metode yang digunakan sangat bervariasi, semuanya memiliki tujuan sama, yaitu mendapatkan representasi sederhana yang berguna dari suatu citra (Castleman, 1996).

2.2.3. Active Contour Model

Konsep *active contours model* pertama kali diperkenalkan oleh M. Kass et al (1987). *Active Contour* menggunakan prinsip energi *minimizing* yang mendeteksi fitur tertentu dalam suatu citra. Sistem ini terdiri dari sekumpulan titik yang saling berhubungan dan terkontrol oleh garis lurus. *Active contour* digambarkan sebagai sejumlah titik yang berurutan satu sama lain. Penentuan obyek dalam citra melalui *active contour* merupakan proses interaktif, sehingga *contour* akan tertarik kearah fitur didalam citra atau *image* karena pengaruh energi internal yang menghasilkan gambar yang ditunjukkan pada Gambar 2.1.



Gambar 2.1. Simulasi *Active Contour Model*

Parameter *active contour* untuk sekumpulan titik koordinat yang terkontrol pada *contour* dapat didefinisikan pada persamaan 2.2 berikut ini :

$$v(s) = \quad x\ s\ , y(s) \tag{2.2}$$

Representasi dan implementasi *active contour* dapat berupa *parametric* atau *geometric*. *Parametric deformable model* direpresentasikan secara eksplisit sebagai kurva yang terparameterisasi dalam ruang Lagrangian, sedangkan *geometric deformable model* direpresentasikan secara *implicit* sebagai *level set* dari fungsi dua dimensi yang berevolusi dalam ruang Eulerian (Kass et al, 1987).

2.2.4. Level Set Method Local Image Fitting Energy

Osher dan Fedkiw (2002) menyatakan bahwa metode *level set* sebagai penggerak kurva. Dalam metode *level set*, kurva ditampilkan secara implicit sebagai *level set* dari fungsi *scalar* 2D dan dikenal sebagai fungsi *level set* yang didefinisikan pada domain yang sama pada citra (Osher et al, 2002).

Adapun perkembangan *level set method,* salah satunya yang dikembangkan oleh Zhang et al (2009). Zhang et al menyatakan bahwa formulasi baru dari fungsi *level set*, yang dapat meminimalisasi energi antara *fitted image* dan citra aslinya. Sehingga algoritma yang dikembangkan oleh Zhang et al disebut *Active Contour Model with Local Image Fitting Energy*. Berikut algoritma *Active Contour Model with Local Image Fitting Energy*, yaitu :

1. Inisialisasi fungsi *level set* ϕ

   Fungsi *levet set* ϕ dinyatakan pada rumus 2.3

$$\phi\ x, t = 0\ = \begin{matrix} -\rho,\ x\epsilon\Omega_0 - \partial\Omega_0 \\ 0,\quad x\epsilon\ \partial\Omega_0 \\ \rho, x\epsilon\Omega - \Omega_0 \end{matrix} \tag{2.3}$$

Dimana $\rho > 0$ merupakan konstanta, Ω_0 adalah bagian dari domain citra Ω dan $\partial\Omega_0$ adalah boundary dari Ω_0.

2. Fungsi *Heaviside* dan Fungsi *Diract*

Berikut merupakan Fungsi *Heaviside* yang dinyatakan dengan H pada persamaan 2.4 dan fungsi *Diract* yang dinyatakan dengan δ pada persamaan 2.4.

$$H_\varepsilon\ \phi\ = \frac{1}{2}\ 1 + \frac{2}{\pi}\arctan(\frac{\phi}{\varepsilon})$$
$$\delta_\varepsilon\ \phi\ = \frac{1}{\pi}.\ \frac{\varepsilon}{\varepsilon^2+\phi^2},\ \phi\epsilon R \tag{2.4}$$

Parameter ε akan memberi efek pada $\delta_\varepsilon(\phi)$. Semakin besar nilai ε maka akan memperluas jangkauan daerah *contour*, tetapi *contour* yang didapat kurang akurat.

3. Fungsi *Rectangular Window*

Fungsi *rectangular window* dinyatakan pada fungsi 2.5 yang berfungsi merata-ratakan intensitas citra pada area dalam dan luar *contour*.

$$m_1 = \frac{K_\sigma *\ H_\varepsilon\ \emptyset\ I(x)}{K_\sigma * H_\varepsilon\ \emptyset}$$
$$m_2 = \frac{K_\sigma *\ (1-H_\varepsilon\ \emptyset\ )I(x)}{K_\sigma *(1-H_\varepsilon\ \emptyset\ )} \tag{2.5}$$

Dimana σ merupakan standar deviasi dan merupakan parameter penting. Apabila nilai σ terlalu kecil memungkinkan hasil yang diperoleh tidak sesuai

dengan keinginan, dan sebaliknya apabila nilai σ terlalu besar akan membutuhkan proses komputasi yang besar.

4. Fungsi *Local Image Fitting Energy*

   Fungsi *Local Image Fitting Energy* dinyatakan pada fungsi 2.6 yang berfungsi meminimalkan perbedaan antara *fitted image* dan citra original.

$$I^{LFI} = m_1 H_\varepsilon\ \emptyset\ +\ m_2(1 - H_\varepsilon\ \emptyset\ ) \tag{2.6}$$

5. Fungsi *Update* $\emptyset^{n+1}$

   Fungsi *update* nilai *level set* ϕ yang dinyatakan pada fungsi 2.7.

$$\phi^{n+1} =\ \phi^n + \Delta t\ \ I - I^{LFI}\quad m_1 - m_2\ \ \delta_\varepsilon(\phi) \tag{2.7}$$

   Dimana n merupakan iterasi, dan Δt merupakan *time-step*.

6. Regularisasi fungsi *level set* dengan kernel Gaussian

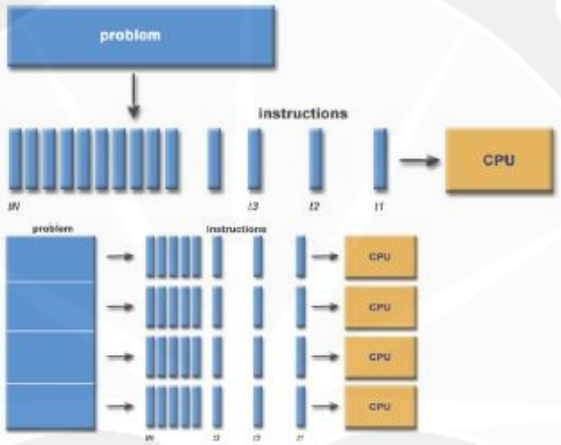
   Fungsi *level set* diregularisasikan dengan kernel Gaussian seperti pada persamaan 2.10.

$$\phi = G_\varsigma * \phi \tag{2.8}$$

   Dimana ς merupakan standar deviasi.

2.2.5. Komputasi Paralel

Proses komputasi paralel merupakan proses komputasi di komputer dengan menggunakan suatu bahasa pemrograman yang dijalankan secara paralel pada saat bersamaan. Secara umum komputasi paralel diperlukan untuk meningkatkan kecepatan komputasi bila dibandingkan dengan pemakaian komputasi pada CPU.



Gambar 2.2. Komputasi Paralel (Ghorpade et al, 2012)

Komputasi paralel adalah perhitungan komputasi dengan menggunakan 2 atau lebih CPU/*processor* dalam suatu komputer yang sama yang ditunjukkan pada Gambar 2.2. Kinerja dari komputasi paralel ini adalah setiap instruksi dibagi kedalam beberapa instruksi kemudian dikirim ke *processor* yang terlibat pada proses komputasi dan dilakukan secara bersamaan. Untuk alokasi memori proses komputasi menggunakan bantuan CUDA *programming*.

2.2.6. GPU (Graphic Processing Unit)

2.2.6.1. Pengertian GPU

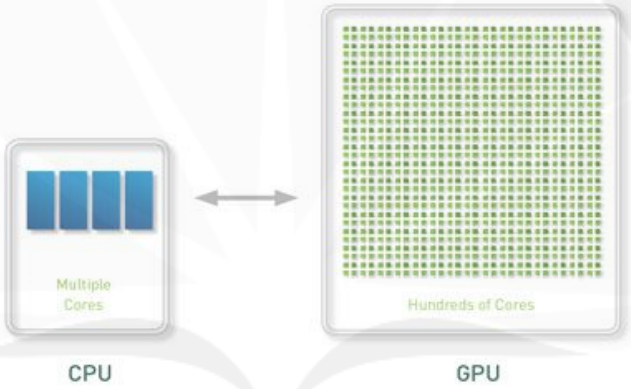
GPU adalah unit pemrosesan grafis yang dapat menjalankan kinerja grafis yang tinggi pada PC, yang merupakan kebutuhan komputasi modern. Pada masa proses evolusi GPU, fungsi GPU sangat terbatas, yaitu mempercepat kinerja grafis. Setelah mencapai titik puncak evolusi, GPU dapat digunakan dalam proses komputasi paralel, dimana GPU menggunakan beberapa *core* yang terdapat pada GPU.

Menggunakan GPU untuk memproses atau menjalankan instruksi non-grafis disebut sebagai *General Purpose Graphic Prcessing Unit*. GPGPU digunakan untuk menjalankan operasi matematika yang kompleks secara parallel dengan kebutuhan waktu yang rendah. Kinerja GPU untuk memperoleh komputasi parallel dengan merubah data skalar ke data vektor. Data vektor dapat menjalankan lebih dari satu instruksi per siklus. Jadi data vektor dapat dihitung secara parallel. Oleh karena itu, GPU dapat bekerja secara parallel.

Komputasi GPU adalah penggunakan GPU untuk tujuan ilmiah dan rekayasa komputasi. Model komputasi GPU adalah menggunakan CPU dan GPU secara bersama-sama dalam satu model komputasi *heterogeneous co-processing*. (Ghorpade et al, 2012)

2.2.6.2. Perbandingan CPU dan GPU

CPU adalah *processor single-chip*. Pada Gambar 2.3 menunjukkan GPU memiliki jumlah *core* lebih banyak dibandingkan dengan jumlah *core* CPU. Pada umumnya jumlah *core* pada CPU adalah 4, sedangkan pada GPU terdapat ribuan *core*. Pada Tabel 2.1 menunjukkan perbandingan antara CPU dan GPU dimana proses kerja GPU lebih cepat dibandingkan dengan CPU. .



Gambar 2.3 Perbandingan *core* CPU dan GPU

(Ghorpade et al, 2012)

Tabel 2.1 Perbandingan CPU dan GPU

No.	CPU	GPU
1.	Kinerja *cache* pada CPU yang cepat	Banyak unit untuk perhitungan
2.	Percabangan pada *granularity* yang baik	Menjalankan program pada setiap lapisan (*vertex*)
3.	Kinerja pada *thread* tunggal	Kinerja secara parallel
4.	Kinerja CPU digunakan untuk *task* secara parallel	Kinerja GPU digunakan untuk data secara parallel
5.	CPU dioptimalkan pada *sequential code*	GPU dioptimalkan pada *floating point operations*

2.2.7. NVIDIA CUDA (Compute Unified Device Architecture)

2.2.7.1. Pengertian CUDA

Evolusi GPU selama bertahun-tahun teah mencapai titik puncak yang diinginkan. NVIDIA memperkenalkan arsitektur komputer parallel secara masal yang disebut "CUDA" dan mengubah sudut pandang pemrograman GPGPU sebelumnya (Ghorpade et al, 2012).

CUDA singkatan dari *Compute Unified Device Architecture* merupakan arsitektur komputer pararel yang dikembangkan oleh NVIDIA. CUDA adalah *platform* komputasi parallel dan model pemrograman yang memungkinkan peningkatan dalam kinerja komputasi dengan memanfaatkan kelebihan dari GPU (NVIDIA, 2006).

2.2.7.2. Arsitektur CUDA

Pada Gambar 2.4 menunjukkan arsitektur CUDA pada dasarnya terbagi menjadi 2 bagian utama, yaitu :

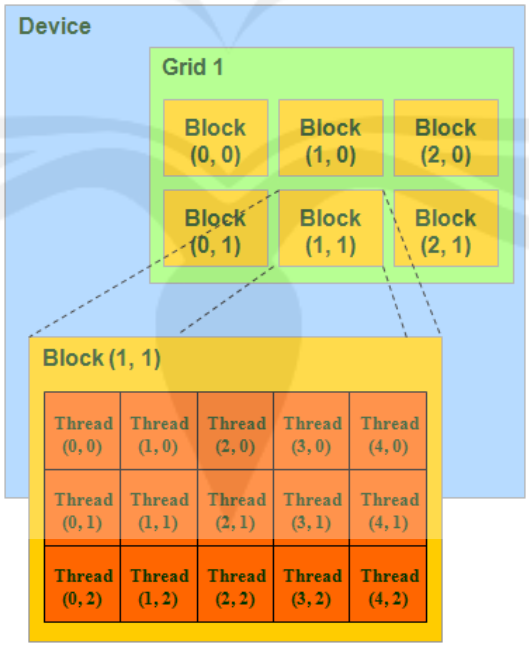
1. *Thread*

   Thread berbentuk sebuah *block* dan dapat digunakan dalam bentuk 1D, 2D dan 3D.

2. *Block* atau Grid

   Block terdiri atas beberapa *thread* di dalamnya. *Block* dapat digunakan dalam skema 1D dan 2D.

Dalam proses komputasi terdapat proses antrian data yang disebut sebagai *warp*. *Warp* memiliki efek pada kinerja GPU, sehingga jumlah *block* dan *thread* yang digunakan memiliki efek pada kecepatan proses komputasi.



Gambar 2.4 Dasar Unit CUDA (Ghorpade et al, 2012)

Pada Gambar 2.5 menunjukkan struktur *memory* pada CUDA, dimana CUDA memiliki beberapa macam *memory* yang dapat digunakan untuk proses komputasi, yaitu :

1. *Global Memory*

   Global memory memiliki sifat sebagai berikut :

   a. Dapat melakukan proses *read* atau *write* dari *thread* ke *block*
   b. Ukuran *memory* cukup besar
   c. Cara kerja *memory* adalah *sequential* sehingga *global memory* pada GPU memiliki kecepatan 10x lebih lambat dibandingkan dengan *memory* yang lainnya.

2. *Shared memory*

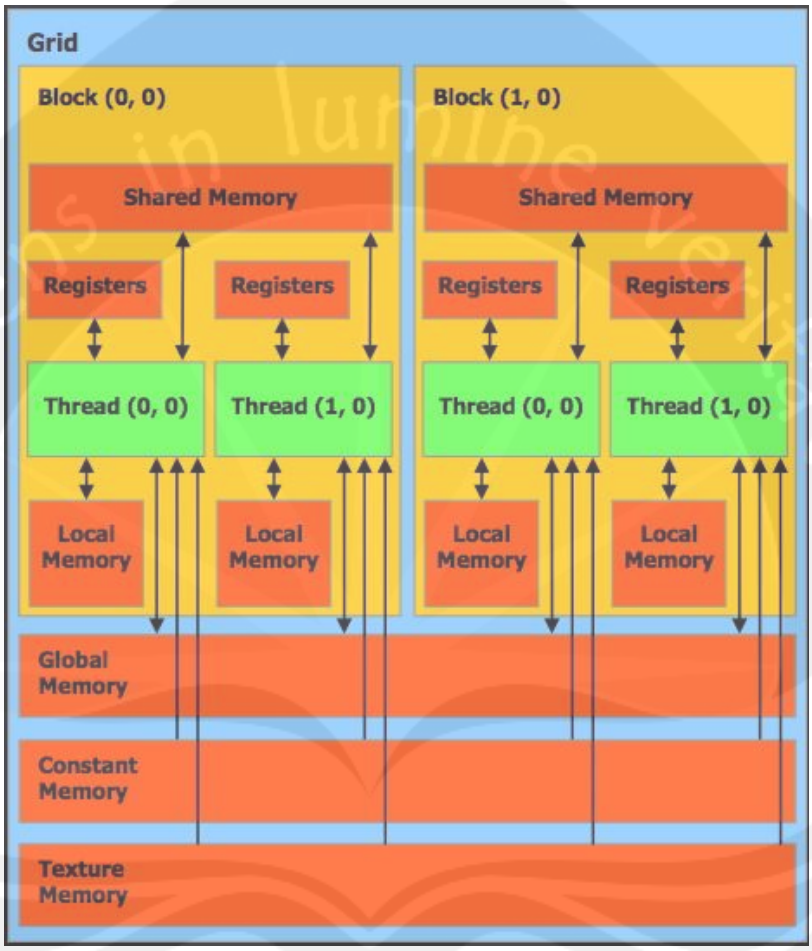
   Shared Memory memiliki sifat sebagai berikut :

   a. *Shared memory* merupakan *memory* yang memiliki kecepatan komputasi paling cepat.
   b. *Shared memory* dapat bekerja lebih cepat apabila membaca data pada *thread* yang sama dengan lokasi *shared memory*
   c. Semua index *shared* array diperoleh dari permutasi

3. *Texture Memory*

   Texture Memory merupakan *memory* yang memiliki proses komputasi yang cepat tetapi hanya dapat mengerjakan proses dalam bentuk 2D.

4. *Register*
5. *Local Memory*
6. *Constant memory*



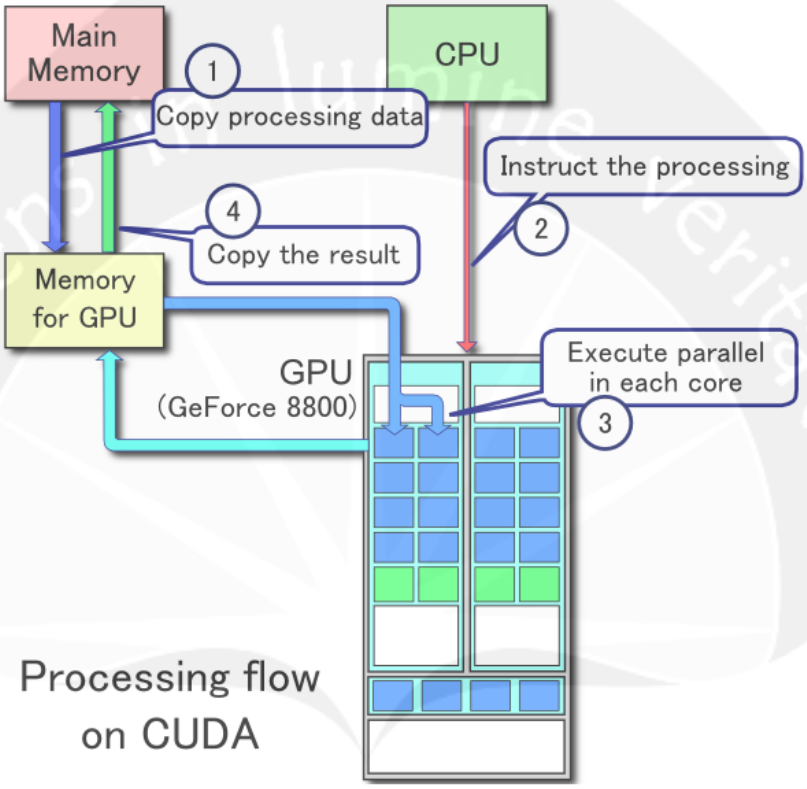
Gambar 2.5 Struktur *Memory* GPU CUDA(Ghorpade et al, 2012)

2.2.7.3. Model Dasar Pemrograman CUDA

Pemrograman berbasis GPU CUDA memiliki langkah- langkah yang berbeda dengan pemrograman lainnya. Pada Gambar 2.6 menunjukkan alur pemrograman CUDA yang memiliki beberapa langkah utama, yaitu :

1. *Copy* data dari *host* ke *device*

2. Proses komputasi sesuai dengan instruksi yang diberikan
3. Menjalankan proses komputasi pada *thread* dan *block* yang ditentukan.
4. *Copy* hasil komputasi dari *device* ke *host*.



Gambar 2.6 Alur Pemrograman CUDA

2.2.7.4. Kebutuhan Instalasi CUDA

CUDA *syntax code* hampir sama dengan bahasa C. Agar dapat menggunakan CUDA, adapun kebutuhan-kebutuhan yang diperlukan, seperti :

1. Visual Studio 2010 GPU driver
2. CUDA SDK
3. Library pendukung lainnya, seperti OpenGL.

2.2.8. Microsoft Visual Studio 2010

Microsoft visual studio 2010 merupakan sebuah perangkat lunak yang dapat digunakan untuk melakukan pengembangan aplikasi yang diciptakan oleh Microsoft. Visual studio terdiri atas

a. *Compiler*

   Compiler terdiri atas Visual C++, Visual C#, Visual Basic, Visual Basic.Net, Visual InterDev, Visual J++. Visual J#, Visual FoxPro, Visual SourceSafe.

b. SDK

c. IDE (*Integrated Development Environment*)

d. Dokumentasi

   Dokumentasi pada visual studio pada umumnya adalah MSDN *Library*.

2.2.9. Matlab

Matlab singkatan dari *matrix laboratory* yang merupakan suatu bahasa tingkat tinggi yang digunakan untuk komputasi teknik. Bahasa ini mengintegrasikan proses komputasi, visualisasi dan pemrograman dengan *environment* yang mudah digunakan dengan mengekspresikan masalah dan solusi ke dalam notasi-notasi matematika.

Secara umum matlab digunakan untuk matematika dan komputasi, pengembangan algoritma, akuisi data, pemodelan dan simulasi, pembuatan prototype, analisis data, eksplorasi, visualisasi dan pengembangan aplikasi termasuk GUI.